CS-308-2014

Tyler: Motion tracking of customers and virtual assistance

Team Name: Athena

Team Code: TH03

Team Members:

Anil Shanbhag - 100050082

Pranav Jindal - 100050006

Rahee Borade - 100050028

Sameep Bagadia - 100050003

Table of Contents

1. Introduction

In this report, we introduce our system **Tyler**, a next-generation system used to replace conventional brick and mortar stores manned by people with virtual assistants controlled remotely. Tyler removes the need for people to be physically present to address customer's questions and allows them to speak remotely sitting at a different place via video conferencing.

Tyler works on principle of 'call center' remotely controlling virtual robots. A set of robots are deployed in the store, each is an equivalent of a sales staff. Robots move around the maze based on input of people density in the store area. Whenever a customer seeks help from the bot, a person sitting remotely in a 'call center' engages the customer.

We see the following use cases for our system:

1. Consider a conventional supermarket where you have staff moving around to help people with their shopping. Tyler can be used to eliminate the need for humans to be present right there, instead a smaller number of people sitting at a 'call center' can attend to customers.

2. Consider a specialty store like a musical instruments store. Customers might want to talk to someone specialist who knows the instruments well. Currently, it is necessary for every store to employee a specialist and this specialist needs to be present round the clock. Using Tyler, we can use the same specialist to attend to many customers and this specialist can attend to people from the comfort of his home.

3. Special help: Sub-parts of Tyler can be used to replace the need for repeated errands of servicemen. Consider for example someone owns a set of aquariums and there is some issue. Having Tyler deployed, the service center staff can remotely solve the problem without having the need to go to the client location.

Motivation

A large number of benefits listed below was the main **motivation** behind our project.

1. Lesser number of people required: The same set of staff in call center engage people in different stores, we anticipate that lesser number of people are required.
2. No need for physical presence.
3. Number of skilled staff needed is reduced.
4. Load-based optimizations: Number of people hired in a particular shift depends on the sum of number of people who visit during this shift.

2. Problem Statement

Implement a proof of concept system for a virtual store. The virtual store should mimic the experience a customer has in a real store. Imagine a customer enters the store and goes to particular rack to checkout a product. A sales person usually is available nearby in case he needs help. The customer should be able to interact with the sales person. We define interaction as the ability to see and ask a doubt to a person and get a response which solves the issue. In the virtual store, the sales person will be bots and these bots should implement functionality for enabling interaction with the customer. The implementation should also move the bots aka sales person around the store automatically without human intervention.

Implementation Details

To send the bots around the store automatically, it is necessary to track people moving around the store. We use a set of overhead cameras (ie: positioned on the ceiling) and using the feed from these cameras to automatically move the bot towards the region where there are customers. Customers can then avail the facility of virtual assistance from assistants sitting at remote locations using video conference. We use a Android phone with WebRTC capabilites, mounted on the Firebird bot for this.

The product should tackle three main problems:

1. **People and Bot tracking**: People and bot tracking is based on cameras positioned across the area under surveillance. It uses a motion tracking computer vision algorithm to track individuals and color based object detection to track bots using these cameras. The tracking will allow us to send the virtual sales persons (bots) to the required locations automatically.

2. **Automatic Bot Movement**: Virtual Assistant is a bot, which can talk to people. The bot should automatically move to locations where the density of customers is more so that customers can avail the virtual assistance. The host uses the camera feed to find the set of people in the camera view, computes the centroid (since there is only 1 camera at the moment) and sends the bot to that location.

3. **Virtual Assistan**t: Implement a dashboard via which an actual person who is sitting behind a computer can remotely engage people via these bots when asked. The virtual assistant should implement a video conferencing solution by which the person behind camera can see and talk to people in the store via the virtual assistant.

3. Requirements

3.1 Functional Requirements

Product Functions

1. **Monitor and detect people**: We detect the positions of people in a given area using the video from the overhead camera.
2. **Provide virtual assistance through video/audio communication through the bot**: We provide a video calling interface via which the customer can interact with the call-centre employee and get the required information.

3.2 Non-Functional Requirements

1. **Availability** : Scheduling requests in such a way as to minimize waiting time of customers and ensure fairness,leading to increase in the system's reliability.

2. **Maintainability** : The system should continuously improve by learning from the past and keeping track of statistical data such as sections most frequented by customers.

3. **Response time** : The communication between the bot and customer should be real time.

3.3 Hardware Requirements

1. 1x Firebird V : These form the base for the virtual assistant.

2. 1x Mountable camera: The camera will be used to track the arena

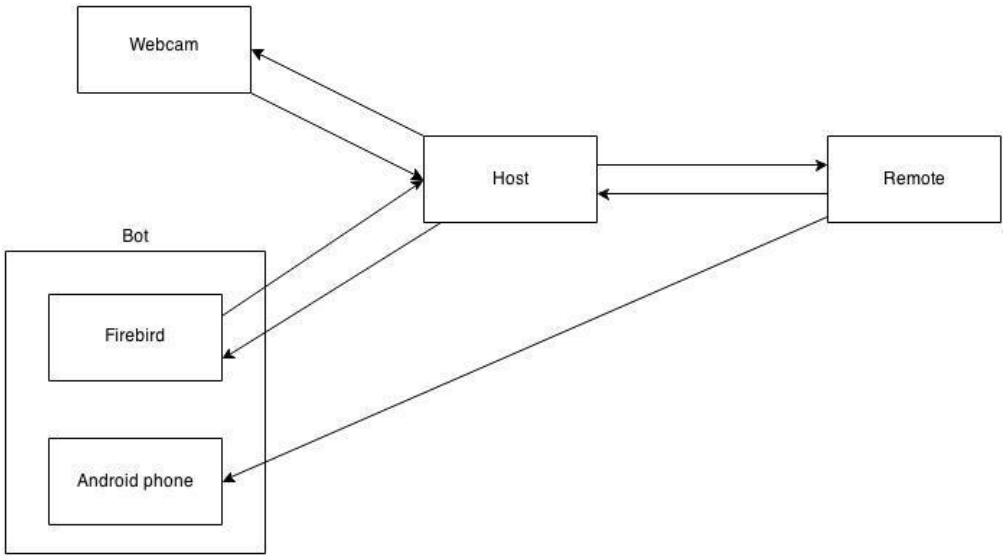
3. 1x Android phone: Each Firebird V will have a android phone. We will implement video/voice communication via an Android App

3.4 Software Requirements

1. Python, including numpy and scipy packages
2. OpenCV package for image processing
3. Keil uVision4 for compiling and building the bot code
4. Websocket for communication via internet
5. WebRTC for realtime audio video communication

4. System Design

Product Perspective

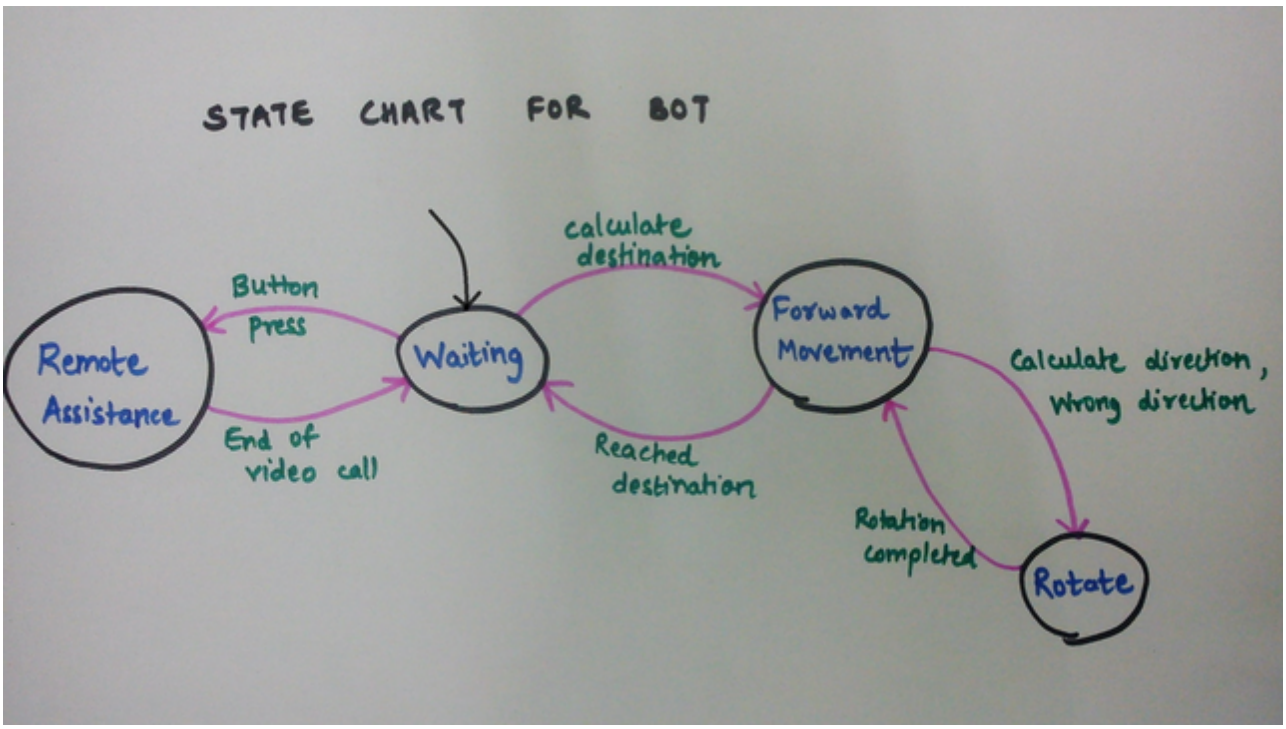


The Tyler system consists of 4 parts:

1. **Bot**: The bot itself consists of two parts. The Firebird V bot used as a base and an android phone is used for video communication between the customer and the call centre employee.
2. **Host**: The Host computer, which is present on the site.
3. **Remote**: The Remote represents the ‘call center’ center employee who will respond to the queries of the customers
4. **Webcam**: A grid of overhead camera is used to track the arena.

The bot can be seen as an independent product. Remote computers are used to connect to the bot. The data collected from the camera is used to guide the guide the bots. The host computer does the processing of feed from webcams. It also acts as a router between the bots and the remote. Real people sit at remote to respond to people queries.

State Chart (FSM)



1. Initially the bot is in the waiting state. Host then calls the Image processing module to get the position of the bot as well as the destination based on the people tracking.
2. Bot initially moves forward to find its current orientation
3. Depending on the current orientation and final destination, host calculates the angle by which it needs to rotate to make the orientation of the bot point towards the destination.
4. If the angle required to rotate is less than a threshold then host directs the bot to rotate by that angle.
5. Whenever the bot is moving forward, we constantly check if it has reached the destination within a certain threshold. If it has reached the destination then it is again in the waiting state where it will wait for few seconds before calculating new destination.
6. When in waiting state, customer can press the button on the bot to initiate remote assistance.
7. During remote assistance, customer can take assistance from remote assistant via video conference.
8. After video call ends, the bot goes back to waiting state.

5. Working of the System and Test results

Tyler implements the following core functionalities. The testing strategy for each functionality has been written directly below it for brevity.

- **People Tracking using Cameras**: A set of low-end cameras attached to the roof are used to monitor the arena. When a person enters the arena, a new person object is created. After regular intervals (~ 100ms) we capture key frames from each camera, run our algorithm to detect the new positions of the people in the arena given their previous positions and finally update the position of each person to the current position. This processing happens in the host system.

  Testing Strategy and Results: For our implementation uses one camera, which was placed on the ceiling, in an empty room. A GUI interface displays the person who enters the room as a red dot. As the person moves around, the dot position will move. We tested this by moving small objects below the camera to be recognized as people by the camera and it worked well.

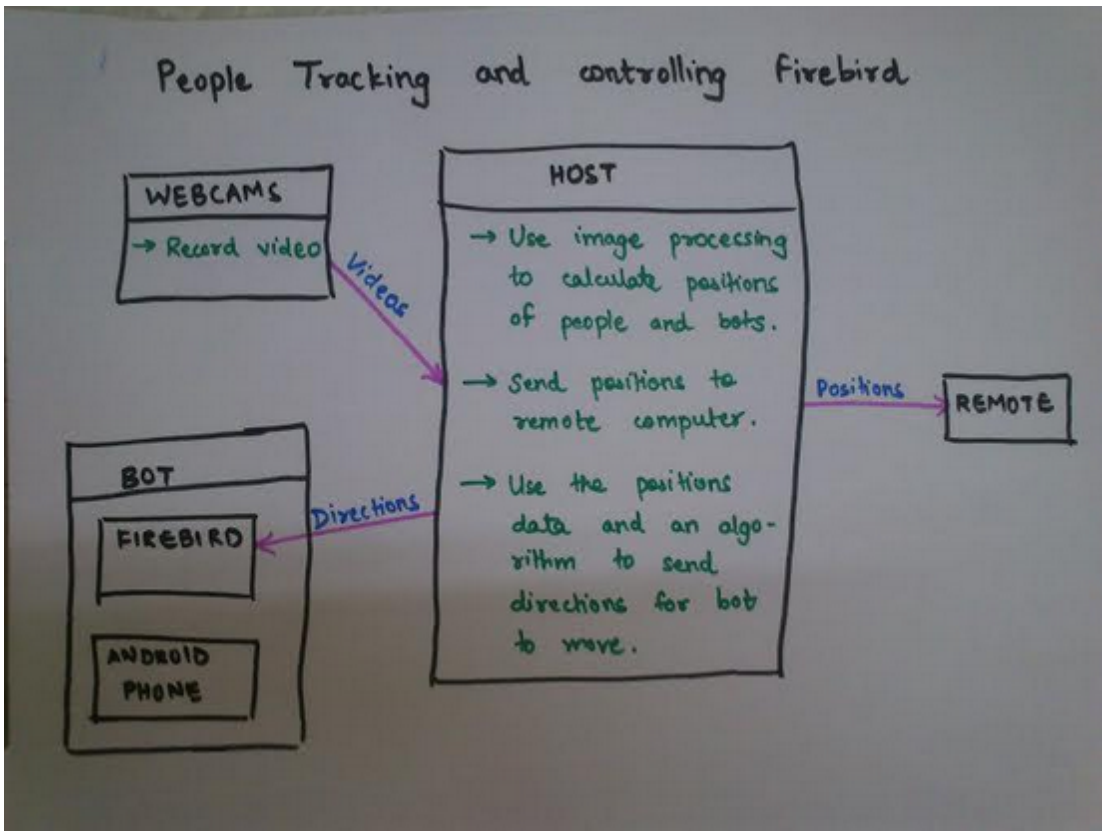


Figure 1: People tracking and controlling Firebird

- **Automatic Bot Movement using Zigbee:** The Firebird V robot forms the base of the virtual assistant and is used for navigation in response to commands. We have implemented communication between the host computer and the virtual assistant using the zigbee. The image processing module gives the location of the bot and the destination it should reach. Bot movement module in the host controls the bot and makes it move to the the destination as required.

Testing Strategy and Results: We give a destination for the bot to go. Using the help of cameras the bot was able to reach the destination and stop as required.

The points 1&2 have been clubbed and information flow has been shown in Figure 1.

- **Communication Interface with Android:** An android phone/tablet sitting on top of the Firebird V is used for audio/video communication between the operator and the customer. There is a small input button on the virtual assistant called "Help", when clicked an operator will engage the person by initiating a audiovideo chat. The phone will connected to the remote via Wi-Fi. Once done, the operator remotely closes the video and the operator is free to attend to other calls.

  Testing Strategy and Test Results: The Firebird V has a mounted android phone. We showcase this functionality by allowing the operator to initiate the video chat once help is asked. This is implemented in the GUI. Video call was real-time and it worked fine.

The information flow of the 3rd component is shown Figure 2.

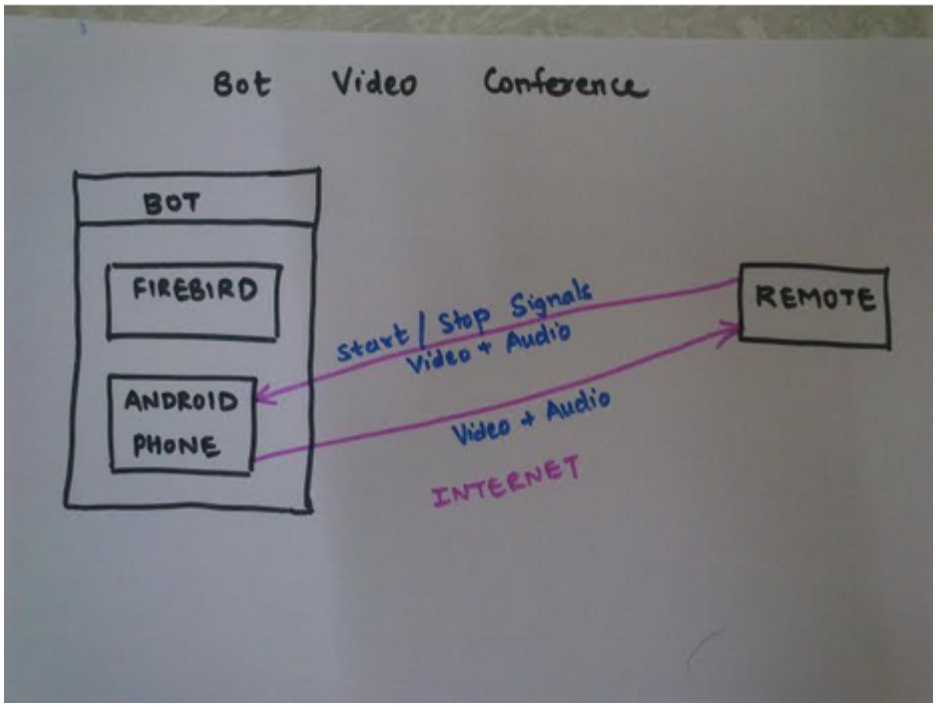


Figure 2: Audio/Video interaction with Bot

6. Discussion of System

a) What all components of your project worked as per plan?

- Image processing to get positions of people and bot
- Movement of bot to the destination
- Button press on the bot to notify need of assistance
- Video calling interface

b) What we added more than discussed in SRS?

- Always on video call leads to battery on the phone quickly drained quickly. So we implemented a manual start/stop on the operator dashboard via which the operator can manually start and stop the video communication with the android phone.
- We used a nice technique to figure out the orientation of the bot. Anytime the orientation is unknown, for example after rotation, we move the bot slightly forward for some cycles and then based on the motion figure out the angle. We used color-based object detection for bot detection.
- Serialport communication via Javascript: The websocket communication and serial port communication are both blocking and hence we were unable to get them to work together in python. We implemented the whole system again in javascript. Since javascript is event-driven, both can work together.

c) Changes made in plan from SRS:

- The bot currently just moves to the centroid of all the moving objects. In the SRS, we had planned to have multiple bots and move each one to particular cluster. We have thought of algorithm for doing this. We can use k-means algorithm with the bots also added as objects. The clustering has a constraint that each cluster should have one bot.

7. Challenges

The project involved a number of different technical challenges:

1. **Localization:** We used an overhead camera to locate the bot. But differentiating the bot from customers was a challenge. We solved this by giving a specific color to the bot and hard-coding the range of HSV values for that color using HSV-masking.

2. **Orientation Detection:** Once the bot was located, finding its orientation was non-trivial. We overcame this by making the bot move in the forward direction and recording the motion vector to indicate its direction.

3. **Multiple blocking calls and their interaction:** On the host side there are 3 things happening:

finding people/bot in the video, zigbee communication with bot and websocket communication with server. Zigbee and websocket communication do not interact so well as both are blocking. We had to rewrite our implementation and do everything all over in javascript.

4. **Audio/Video communication:** Implementing a good audio/video communication system from scratch is challenging. We thought of simply using Skype but then we finally chose to implement using WebRTC which is supported by most new browsers.

8. Known bug report

While using the computer vision algorithms we face the following shortcomings:-

1. The tracking of people via motion detection is susceptible to noise in the form of moving shadows and instability of the camera. There is a need to remove shadow and effect of reflection from the video stream.
2. Detecting the bot using a mask of HSV values requires hard coding of the range of HSV values, which is dependent on the illumination conditions.

Over the internet communication using WebRTC:-

1. WebRTC used for video communication works nicely when there are no firewalls. This is true if we are doing everything inside local network or there is no firewall in between. The presence of firewall will lead to failure. To prevent this, we need to setup a TURN server, however this was not done as both host and remote were on local network.

9. Reusable components

We implemented the following reusable components:-

Image Processing modules

1. Detecting an object through it's color: this component finds the position of an object in the image frame, having a specific color using HSV-masking.
2. Motion tracking for detecting people: Finds all the objects in the image frame which have moved since the previous image frame in a video sequence. The main steps in the algorithm are :-
   2.1. **Detecting Motion:** we first create a running average of the incoming video frames, of the last ~0.25 seconds worth of frames (OpenCv provides RunningAvg()). To search for motion, we then take the current camera frame and subtract it from the running average, using OpenCV's AbsDiff() function
   2.2. **Detecting Moving Blobs:** To find the "blobs" in the new black-and-white difference

image,we use the OpenCV function FindContours(), followed by ApproxPoly() to get polygon coordinates. That returns a list of coordinates that represent polygons surrounding the white area(s)

2.3. **Detecting Targets:** The solution is a combination of axis-aligned bounding boxes (AABBs), and the k-means algorithm. We used AABBs to convert disconnected blobs around the perimeter of a moving object into a single entity that could be tracked. I found the AABB of each blob, and then merged all overlapping AABBs together recursively. This results in a box that grows in size until it encompasses all of the nearby "puddles" of white motion pixels into a single box.We use the number of recursively merged AABBs to get an estimate of the number of targets. The number of AABBs tells me how many non-overlapping regions of motion are in the frame, which is a rough approximation of how many individual entities there are. Then the k-means function gives a more accurate analysis of the actual number of cluster that were found.

Bot movement

1. We implemented an interface which allows a remote host to control the movement of a bot through the internet.
2. Bot movement was accomplished by sending commands like Forward, Back, Stop, Right, Left, Soft_Right, Soft_Left etc from the remote host to a system in direct connection with the bot (via a Zigbee network).
3. Rotation by Theta: this component allows us to rotate the bot by a given angle theta, which can be used for changing the orientation of the bot.The function Angle_Rotate implements this.

Realtime Audio-Video Communication

1. In our dashboard, we implemented real time audio video communication using WebRTC (www.webrtc.org). WebRTC is available on all latest browsers on the desktop and the Chrome browser on Android. In our project, this is used to enable communication between the mounted Android phone and the dashboard window.
2. Manual Start-Stop: We implemented manual start-stop, so that the operator sitting remotely can start or stop the video call to the phone by pressing a button. This is necessary as phone battery is precious and having an always on video call will not last long

10. Future Work

Possible Extensions to Tyler (Future work)

The project currently implements a single camera, single bot system. However the architecture is built in a manner that makes it scalable. Things that can be done to make the system usable:

1. Add multiple cameras and multiple bots to control.

2. Work can be done in the direction of scheduling the multiple bots to service the large number of customers efficiently.
3. We have currently assumed absence of any obstacles in the path of the bot. Thus, we will also need to store the locations of obstacles like shelves etc in the store and navigate the bot through these.
4. On a visual front: make the bot like R2D2 from star wars.

11. Conclusion

Thus, through our project we demonstrate the viability of the hi-tech stores fully equipped with virtual assisting bots. We show the suitability of using motion tracking for detecting people in the stores, and using this as input for automatic bot scheduling. Finally, we also showed a remote call center modal and its ease of use via the Tyler Dashboard.

12. References

WebRTC http://www.webrtc.org/

Video Conferencing in the browser http://appear.in

Flowtracker algorithm OpenCV 2.4.8.0 documentation, Motion Analysis and Object tracking http://docs.opencv.org/modules/video/doc/motion_analysis_and_object_tracking.html

Android Application http://developer.android.com

Chrome WebView https://github.com/pwnall/chromeview

The data collected is useful, a number of startups doing in-store analytics http://retailnext.net/
Even IBM is joining the space, http://www.ibm.com/smarterplanet/us/en/retail_analytics/ideas/